University of Saskatchewan
Department of Computer Science

Cmpt 340
Midterm Examination

February 22, 2007

Professor: Tony Kusalik
Format: Closed Book

Time: 80 minutes
Total Marks: 81

Name: _____

Student Number: _____

Directions

Answer each of the following questions in the space provided in this exam booklet and hand in the booklet when you are finished. If you must continue an answer (e.g. in the extra space on the last page or on the back side of a page), make sure you clearly indicate that you have done so and where to find the continuation.

Make all written answers legible; no marks can be given for answers which cannot be decrypted. Where a discourse or discussion is called for, please be concise and precise. Do not give "extra answers". Extra answers which are incorrect may result in your being docked marks. If you find it necessary to make any assumptions to answer a question, state the assumption with your answer.

Marks for each major question are given at the beginning of that question. There are a total of 81 marks.

Good luck.

For marking use only:

A. ____/6	D. ____/10	F. ____/12
B. ____/ 8	E. ____/ 5	
C. ____/10	F. ____/30	**Total**: ____/81

A. (6 marks)

Fill in the blanks to make each of the following statements true in the context of the material in Cmpt340. Each correctly filled-in statement is worth 1 mark.

1. In Prolog, the _____ variable is a variable whose name is simply a single occurrence of the underscore character, '_'.
2. The _____ of a compound term refers to the number of subterms in it.
3. A Prolog program is complete if _____ ⊆ _____.
4. A recursive Prolog program in which the recursive call is the last subgoal of the body of rules is called _____ _____.
5. A _____ is a subgoal or set of subgoals which can be used to make multiple clauses mutually exclusive.
6. An example of a _____ interpreter is an interpreter for Prolog written in Prolog.

B. (2+2+2+2 = 8 marks)

Answer each of the following questions with a very short, precise answer.

1. Consider the following Prolog program:

```
% student( StudNo, Name, College, Year, Major )
student(12345, adelia, ar, 2002, undeclared).
student(38475, bradford, ar, 1995, phil).
student(92845, cao, fgsr, 2001, eng).

% class( ClassId, Dept, CourseNum, Session, Term )
class(73463, cmpt, 115, s, t2).
class(47312, cmpt, 317, w, t2).
class(98457, engl, 100, s, t1).

class(16384, phys, 488, w, t2).

% enrolled( StudNo, ClassId )
enrolled(12345,73463).
enrolled(12345,16384).
enrolled(38475,47312).
enrolled(38475,98457).
enrolled(92845,73463).
enrolled(92845,98457).
```

   What Prolog query seeks a solution to the question "Who is enrolled in both Cmpt 115 and Engl 100?" Give such a query.

2. Consider the Prolog term

```
assembly( bike, [ wheel, wheel, frame ], 3+4*5 ).
```

   Give four functors used/present in the above term.

3. Give a more convenient syntax for the list `.(f,.(g,.(f(g),Xs)))` which makes use of the list shorthand supported by the Prolog parser.

4. Give the name of two Prolog programming approaches which can be used to dramatically improve the performance of Prolog programs without sacrificing correctness. "Green cuts" is not one of the answers.

C. (6+2+2 = 10 marks)

Answer each of the following questions with a short, technically precise answer.

1. Consider the two Prolog terms

```
assembly( What, [wheel,wheel,Missing], L )
```

and

```
assembly( bike, [First,_|Rest], X )
```

i) Give the most general unifier of these two terms.

ii) Give a substitution for the second `assembly/2` term above which violates the occurs-check condition. Note that the would-be substitution is for the second term only.

iii) Give a unifier for the above two terms which is not a most general unifier.

2. Recall the steps in the algorithm for Prolog resolution given in class. The first two steps were to (1) remove a goal from the resolvent and (2) to unify the goal with the head of a fact or rule from the program. What are the next two steps?

3. Recall the standard definition for the predicate for appending two lists given in class:

```
append( [], L, L ).
append( [X|L1], L2, [X|L3] ) :- append( L1, L2, L3 ).
```

This definition of `append/3` is not correct. In particular, there are queries in *M(P)* which are not in *M*. Give one such query.

D. (10 marks)

Insertion sort is a sorting algorithm which accumulates a sorted list from its input list by repeatedly and consecutively inserting the elements of the input list into the partial sorted list in their correct positions. For example, given the input list [4,1,5,3], the algorithm proceeds as follows

input list (partial)	sorted list
`[4,1,5,3]`	`[]`
`[1,5,3]`	`[4]`
`[5,3]`	`[1,4]`
`[3]`	`[1,4,5]`
`[]`	`[1,3,4,5]`

Implement insertion sort in Prolog. Define `ins_sort/2` as well as an auxiliary predicate `insert/3`. Assume that the program only needs to work with lists of integers and that `ins_sort/2` predicate will have usage pattern `(+,?)`. The input list (the first argument) may contain duplicates. Recall that the built-in

Prolog predicate for "less than or equal to" and "greater than or equal to" are =< and >=, respectively. Marks will be given for clarity and elegance.

E. (5 marks)

A student has written the following Prolog program

```
% factorial(+N, ?NFactorial)
% NFactorial = N * (N-1) * (N-2) * ... * 2 * 1

factorial(0,1).
factorial(N,F) :- N > 0, factorial(N-1, F1), F is N * F1.
```

Unfortunately, a quick test indicates to the student that the program has a bug in it. E.g.

```
| ?- factorial( 4, X ).
no
```

The student decides to following a declarative debugging strategy to find (and fix!) the bug. The following is a log of the student's progress through the declarative debugging procedure:

```
| ?- trace, factorial( 4, X ).
% The debugger will first creep -- showing everything (trace)
        1      1 Call: factorial(4,_454) ? s
        1      1 Fail: factorial(4,_454) ? r
        1      1 Call: factorial(4,_454) ? c
        2      2 Call: 4>0 ?
```

What should the student do now? I.e. according to the declarative debugging procedure, what should the next few steps or actions be?

F. (5+2+8+3+4+4+4 = 30 marks)

Consider the following Prolog program:

```
disease(flu).                    % clause 1
disease(brokenBone).             % clause 2
disease(cancer).                 % clause 3
disease(hypothermia).            % clause 4
disease(asthma).                 % clause 5

symptom(sneezing).               % clause 6
symptom(pain).                   % clause 7
symptom(headache).               % clause 8
symptom(shivering).              % clause 9

causes(flu,sneezing).            % clause 10
causes(flu,headache).            % clause 11
causes(brokenBone,pain).         % clause 12
```

```
causes(cancer,pain).                    % clause 13
causes(hypothermia,shivering).          % clause 14
causes(asthma,sneezing).                % clause 15

curable(brokenBone).                    % clause 16

relief(S) :-                            % clause 17
       symptom(S), causes(D,S), disease(D), curable(D).
```

1. Give a search tree for the query `?-relief(sneezing)`. In the tree, mark backtrack points with unexplored alternatives with an asterisk ('*').

2. Give the proof tree for the query in Question F.1. If there is not proof tree, say "no proof exists".

3. Give a search tree for the first solution to the query `?-relief(What)`. In the tree, mark backtrack points with unexplored alternatives with an asterisk ('*').

4. Give the proof tree for the query in Question F.3. If there is not proof tree, say "no proof exists".

5. Should the instantiation or usage pattern for goals `relief/1` be `relief(?S)` or `relief(-S)` or `relief(+S)`? Why? I.e. justify your answer.

6. Suppose the definition of the `relief/1` predicate was changed to

```
relief(S) :-
      symptom(S), causes(D,S), !, disease(D), curable(D).
```

Would the introduced cut be a red cut or a green cut? Why?

7. Suppose the definition of clause 8 was changed to

```
symptom(headache) :- !.        % clause 8
```

and otherwise the program was as above. Would the introduced cut be a red cut or a green cut? Why?

G. (5+4+3 = 12 marks)

Answer each of the following questions with a discussion-oriented answer. Feel free to use examples to illustrate your points.

1. What is the difference between don't know and don't care nondeterminism in Prolog? Contrast the two terms with a technically precise explanation. Examples would be an excellent way to illustrate your points.

2. Explain the difference between the computations induced by following two Prolog goals

```
X is 7 + 3.
```

and

```
X = 7 + 3.
```

3. One of the themes in Cmpt340 is that different computer programming languages are good in different application domains or for solving different kinds of problems. Give a type of problem or application for which Prolog is a good choice as programming language. Then explain why Prolog is a good choice for that type of problem or application.